Open for Innovation KNIME

← Blog

Read time: 9 min

4 levels of LLM customization

From prompt engineering to artificial agents

February 28, 2023

Put simply | ML 201 & AI

Open for Innovation KNIME

LLMs, or Large Language Models, represent an innovative approach to enhancing productivity. They have the ability to streamline various tasks, significantly amplifying overall efficiency.

In this article, we want to look at how you can customize LLMs to make them even more useful both day-to-day activities and professional endeavors.

We'll explore four levels of LLMs customization that build on each other:

1. Prompt engineering to extract the most informative responses from chatbots

subjects

4. Agents to construct domain-specialized models

1. Prompt engineering

A prompt is a concise input text that serves as a query or instruction to a language model to generate desired outputs. Put simply, it represents the most straightforward manner for human users to ask LLMs to solve a task.

The lightning-fast spread of LLMs means that crafting effective prompts has become a crucial skill, as the instructions provided to the model can greatly impact the outcome of the system. Good prompt engineering involves creating clear and onpoint instructions in a way that maximizes the likelihood of getting accurate, relevant, and coherent responses.

Well-engineered prompts serve as a bridge of understanding between the model and the task at hand. They ensure that the generated responses directly address the input. Additionally, they play a vital role in reducing biases and preventing the model from producing inappropriate or offensive content. This is particularly important for upholding ethical and inclusive AI applications.

Conversely, a poorly constructed prompt can be vague or ambiguous, making it challenging for the model to grasp the intended task. It might also be overly prescriptive, limiting the model's capacity to generate diverse or imaginative responses. Without enough context, a prompt might lead to answers that are irrelevant or nonsense.

Some of the best practices for good prompt engineering include:

- If possible, pick the latest model (they are likely to be smarter).
- Put instructions at the beginning of the prompt.
- Be specific, descriptive and as detailed as possible about the desired context, outcome, length, format, style, etc.
- If possible, provide examples of the desired output format.

Open for Innovation KNIME

1

Extract name entities from this text: Charles Spencer Chaplin Jr. was born on 16 April 1889 to Hannah Chaplin (née Hill) and Charles Chaplin Sr. His paternal grandmother came from the Smith family, who belonged to Romani people. There is no official record of his birth, although Chaplin believed he was born at East Street, Walworth, in South London. His parents had married four years previously, at which time Charles Sr. became the legal guardian of Hannah's first son, Sydney John Hill.

2023-10-13; 13:38

- Charles Spencer Chaplin Jr.
- Hannah Chaplin
- Charles Chaplin Sr.
- Smith family
- Romani people
- East Street
- Walworth
- South London
- Sydney John Hill

2023-10-13; 13:38

Write your message

Extract name entities from this text: Charles Spencer Chaplin Jr. was born on 16 April 1889 to Hannah Chaplin (née Hill) and Charles Chaplin Sr. His paternal grandmother came from the Smith family, who belonged to Romani people. There is no official record of his birth, although Chaplin believed he was born at East Street, Walworth, in South London. His parents had married four years previously, at which time Charles Sr. became the legal guardian of Hannah's first son, Sydney John Hill.

Send message

2

Extract name entities and assign a name entity tag from this text: Charles Spencer Chaplin Jr. was born on 16 April 1889 to Hannah Chaplin (née Hill) and Charles Chaplin Sr. His paternal grandmother came from the Smith family, who belonged to Romani people. There is no official record of his birth, although Chaplin believed he was born at East Street, Walworth, in South London. His parents had married four years previously, at which time Charles Sr. became the legal guardian of Hannah's first son, Sydney John Hill.

2023-10-13; 13:40

Name entities and their corresponding name entity tags in the given text:

1. Charles Spencer Chaplin Jr. - PERSON
2. Hannah Chaplin - PERSON
3. Charles Chaplin Sr. - PERSON
4. Smith - PERSON
5. Romani - NORP
6. East Street - FAC
7. Walworth - LOC
8. South London - LOC
9. Sydney John Hill - PERSON

2023-10-13; 13:40

Write your message

Extract name entities and assign a name entity tag from this text: Charles Spencer Chaplin Jr. was born on 16 April 1889 to Hannah Chaplin (née Hill) and Charles Chaplin Sr. His paternal grandmother came from the Smith family, who belonged to Romani people. There is no official record of his birth, although Chaplin believed he was born at East Street, Walworth, in South London. His parents had married four years previously, at which time Charles Sr. became the legal guardian of Hannah's first son, Sydney John Hill.

Send message

Open for Innovation KNIME

which could later be used to train a named entity recognizer.

Recently, the rise of AI tools specifically designed to assist in the creation of optimal prompts promise to make human interactions with conversational AI systems even more effective.

While crucial, prompt engineering is not the only way in which we can intervene to tailor the model's behavior to align with our specific objectives. The next level of LLMs customization is hyperparameter tuning.

2. Hyperparameter tuning

LLMs hinge on a complex transformer-based architecture, billions of trainable parameters, and vast datasets to be proficient in the way they think, understand, and generate outputs. These parameters represent the internal factors that influence the way the model learns during training and the quality of its predictions.

In particular, model parameters serve to capture patterns in the data, are automatically adjusted by the mode, and ensure accurate representation of learned patterns. On the other hand, hyperparameters represent the external factors that influence the learning process and outcome.

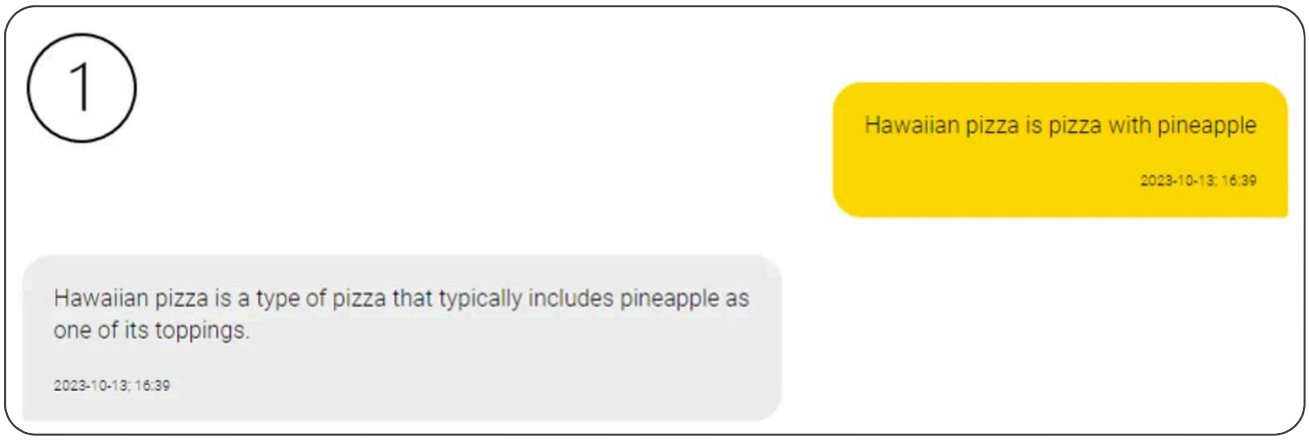
They are a set of configurable options determined by the user and can be tuned to guide, optimize, or shape model performance for a specific task.

Similar to traditional machine learning or deep learning models, in LLMs there exist several hyperparameters to customize the behavior of the model. A few particularly noteworthy ones are temperature, context window, maximum number of tokens, and stop sequence.

Temperature

Temperature ranges from 0 to 2 and serves as a control knob over the level of randomness exhibited in the model's outputs. A higher temperature setting leads to more creative and imaginative responses, while a lower temperature setting results in answers that are more precise and factually grounded.

(see below).

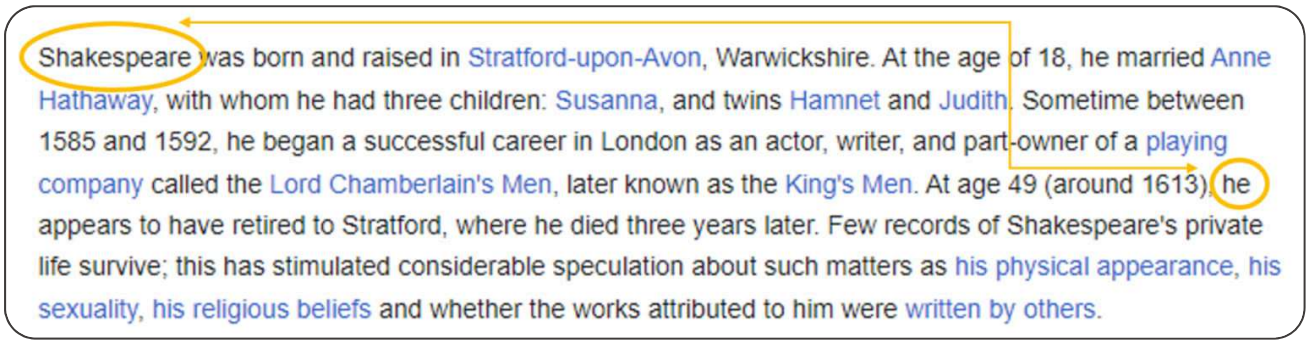


Examples of using different temperature values to let the model complete the text provided in the prompt (yellow bubble). In (1) temperature = 0, whereas in (2) temperature = 1.5.

Context window

The context window defines the number of preceding tokens (words or subwords) that the model takes into account when generating text. A larger context window empowers the LLM to craft responses that are more contextually attuned, albeit at the expense of increased computational resources during the training process.

the user can choose to employ models with larger/smaller context windows depending on the type of task at hand.


Shakespeare was born and raised in Stratford-upon-Avon, Warwickshire. At the age of 18, he married Anne Hathaway, with whom he had three children: Susanna, and twins Hamnet and Judith. Sometime between 1585 and 1592, he began a successful career in London as an actor, writer, and part-owner of a playing company called the Lord Chamberlain's Men, later known as the King's Men. At age 49 (around 1613), he appears to have retired to Stratford, where he died three years later. Few records of Shakespeare's private life survive; this has stimulated considerable speculation about such matters as his physical appearance, his sexuality, his religious beliefs and whether the works attributed to him were written by others.

Excerpt of the Wikipedia page about Shakespeare. LLMs with large context windows can easily map long-range dependencies for coreference resolution (i.e. to disambiguate that "he" refers to "Shakespeare").

Maximum number of tokens

The **maximum number of tokens**, on the other hand, refers to the maximum number of tokens that model generates in the output. Setting a specific value for this hyperparameter limits the length of the response generated by the model, preventing excessively long or short output and tailoring the length of the model's responses to suit the use case. It is worth noting that the maximum number of tokens typically includes both the tokens generated by the model and the tokens present in the input prompt. This means that if a rather verbose prompt is used or/and output is desired, this hyperparameter needs to be set to a high enough value in order to meet the requirements.

Furthermore, LLMs can be programmed to steer clear of generating specific sequences, such as profanity or sensitive information. This is achieved by incorporating a designated hyperparameter called **stop sequence**. It is associated with a particular word, set of words or concept and effectively ensures that the model refrains from generating it in future outputs.

In the popular realm of conversational AI (e.g., chatbots), LLMs are typically configured to uphold coherent conversations by employing an extended context window. They also employ stop sequences to sieve out any offensive or inappropriate content, while setting the temperature lower to furnish precise and on-topic answers.

While fairly intuitive and easy, relying solely on prompt engineering and hyperparameter tuning has many limitations for domain-specific interactions. Generalist LLMs usually lack very specialized knowledge, jargon, context or up-to-date information needed for certain industries or fields. For example, legal professionals seeking reliable, up-to-date and accurate information within their domain may find interactions with generalist LLMs insufficient.

Retrieval-augmented generation (RAG) is an AI framework that promises to overcome the limitations above by fine-tuning and enriching the quality of LLM-generated responses with curated sources for specific applications. This means that a company interested in creating a custom customer service chatbot doesn't necessarily have to recruit top-tier computer engineers to build a custom AI system from the ground up. Instead, they can seamlessly infuse the model with domain-specific text data, allowing it to specialize in aiding customers unique to that particular company.

The RAG process involves three steps:

1. **Retrieval.** Retrieve relevant information from a knowledge source based on the input query.
2. **Augmentation.** Augment the input query or prompt with the retrieved information. This enhances the model's understanding by providing additional context from the retrieved sources.
3. **Generation.** Generate a more informed and contextually rich response based on the augmented input leveraging the generative power of the model.

In practice, to enable LLMs to supplement their in-built knowledge, the concept of Vector Stores and Agents comes into play.

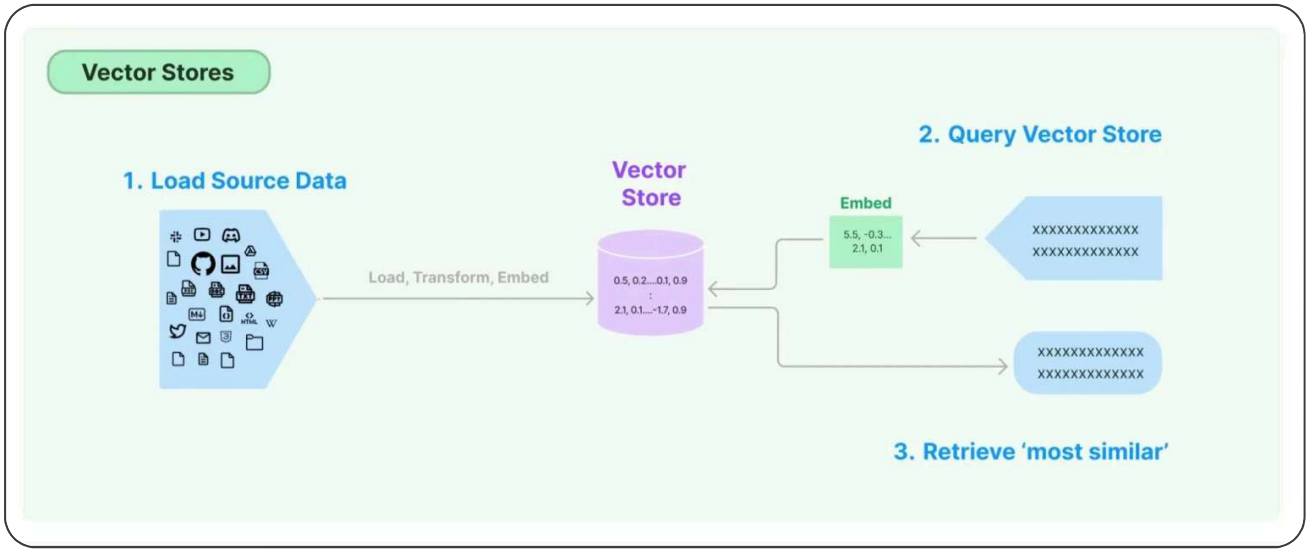
Vector stores

At its core, a vector essentially is an array of numbers. However, what makes vectors truly fascinating is their capacity to represent more intricate entities such as words, phrases, images, or audio files within a continuous, high-dimensional space known as an embedding. These embeddings effectively map the syntactic and semantic meaning of words or shared features in a wide range of data types. They find utility in applications, such as recommendation systems, search algorithms, and even in

Open for Innovation KNIME

Unlike a conventional relational database, which is organized in rows and columns, or a document database with documents and collections, a vector database arranges sets of numbers together based on their similarity. This design enables ultra-fast querying, making it an excellent choice for AI-powered applications. The surge in popularity of these databases can be attributed to their ability of enhancing and fine-tuning LLMs' capabilities with long-term memory and the possibility to store domain-specific knowledge bases.

The process involves loading the data sources (be it images, text, audio, etc.) and using an embedder model, for example, OpenAI's Ada-002 or Meta's LLaMA to generate vector representations. Next, embedded data is loaded into a vector database, ready to be queried. When a user initiates a query, this is automatically embedded and a similarity search across all stored documents is performed. In this way, pertinent documents are retrieved from the vector database to augment the context information the model can rely on to generate tailored responses. Popular vector store databases are Chroma and FAISS.



Schema showing how Vector Stores can customize and enhance LLMs (source: https://python.langchain.com/docs/modules/data_connection/vectorstores/)

The use of RAG is beneficial not only for improving the performance of AI systems (e.g., custom knowledge base, up-to-date information, etc.) but also to mitigate the risk of hallucination, generation of toxic content or biased information.

Open for Innovation KNIME

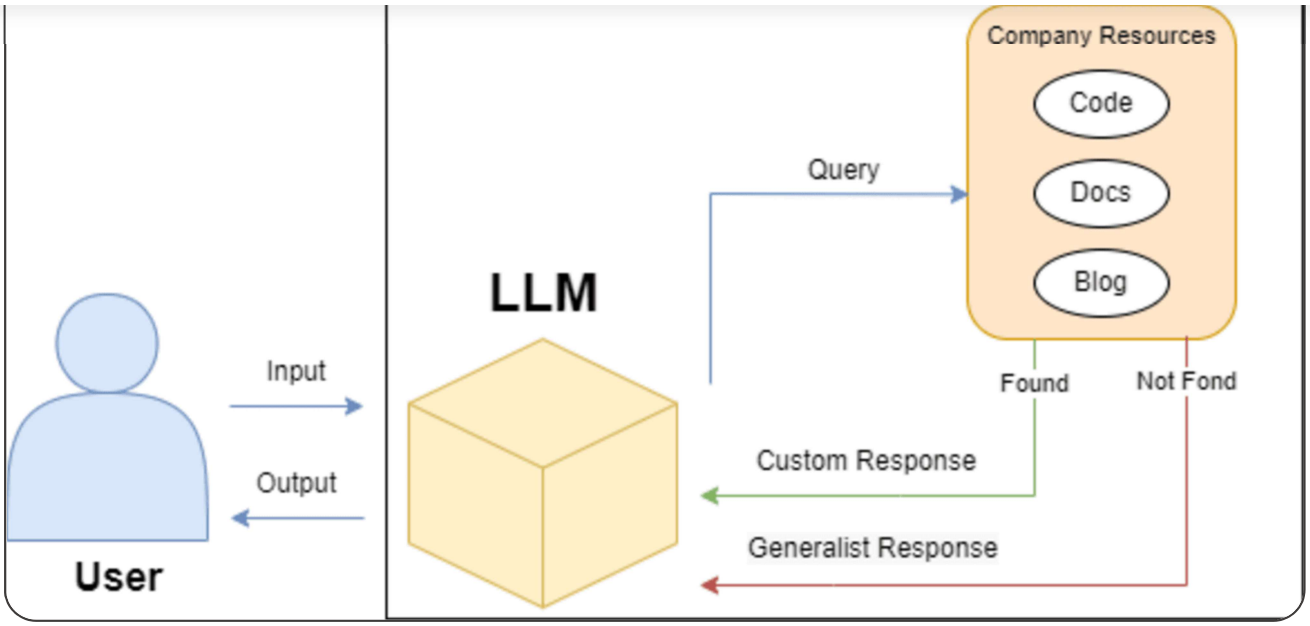
4. Agents

Agents, more precisely referred to as Conversational Retrieval Agents in highly customized LLMs settings, play an essential role in creating tailored conversational AI systems that leverage domain-specific resources for personalized humane-machine interactions. By "agents", we mean a system where the sequence of steps or reasoning behavior is not hard-coded, fixed or known ahead of time, but is rather determined by a language model. This can allow the system greater flexibility and adaptability.

Agents rely on the conversational capabilities of generalistic LLMs but are also endowed with a suite of specialized tools (usually one or more vector stores). Depending on the user's prompt and hyperparameters, the agent understands which, if any, of these tools to employ to best provide a compelling response. Moreover, they can be instructed to perform specific functions or roles in a certain way. For example, an agent can be prompted to write a political text as if it was a poet of the Renaissance or a soccer commentator.

In a medical context, for example, the agent might help physicians treat patients best by leveraging tools for diagnosis, treatment recommendations, or symptom interpretation based on the user's specific inquiry. The incorporation of vector stores on medical literature and instructions to behave as a helpful medical assistant empower the agent with domain specific information and a clear function. This means that the agent not only has access to the generic knowledge base it was trained on but that it can now choose to leverage a rich database of specialized information to help physicians identify the best treatment.

The integration of agents not only makes LLMs versatile but also enhances their capability to deliver tailored outputs specific to a given domain. This specialization ensures that the responses provided are not only accurate but also highly relevant to the user's specific query.



Schema showing how an agent interacts with the vector store database to generate a custom response using the information in the company resources. The schema also shows that when the information queried is not available, the agent reports a generalist response using only vanilla LLM.

Design tailor-made AI

The application of the four techniques discussed in this article enables a deeper level of understanding and utilization of LLMs for productivity.

1. Prompt Engineering gives us a simple way to unlock the potential of these models with just a few instructions.
2. Hyperparameter tuning enables us to fine-tune the model's responses, providing a higher degree of control.
3. Retrieval Augmented Generation gives us a transformative approach that enables LLMs to acquire domain-specific knowledge, greatly enhancing their effectiveness in specialized tasks.
4. The integration of Vector Stores and Agents elevates customization to a whole new level, resulting in context-specific AI systems that deliver tailored, precise, and highly pertinent responses.

Open for Innovation KNIME

Author

Vittorio Haardt

Vittorio is currently interning at KNIME in the Evangelism Team. He's pursuing a Master's degree in Data Science and holds a Bachelor's degree in Statistics and Information...

Sign up to our mailing list

You might also like

BLOG

What are AI hallucinations & how to mitigate them in LLMs

January 8, 2024 | by Roberto Cadili

Open for Innovation KNIME

Read more →

BLOG

An introduction to Large Language Models

February 27, 2024 | by Vittorio Haardt, Swetha Kannan

Read more →

Open for Innovation KNIME

BLOG

Mitigate hallucinations with retrieval augmented generation in KNIME

January 10, 2024 | by Roberto Cadili

Read more →

Open for Innovation KNIME

Software

Overview

Why Visual Workflows

KNIME Analytics Platform

Open for Innovation KNIME

Pricing & Plans

Overview
Team Plan

Get Started

Download
Contact

Solutions

By Industry

Financial Services
Retail & CPG
Manufacturing
Life Sciences
Energy & Utilities
Public Sector
Analytics Consulting

By Department

Marketing
HR
Finance
Supply Chain

By Use Case

Spreadsheet Automation
Continuous Deployment of Data Science
Generative AI

By Role

Data Experts
Business/Domain Experts
End Users
MLOps/IT
Educators

Community

Open for Innovation KNIME

Imprint
Terms of use
Privacy